VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

SKLÁDÁNÍ PANORAMAT Z FOTOGRAFIÍ

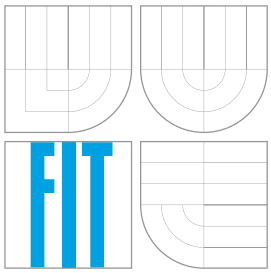
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

LUKÁŠ POSPÍŠIL

BRNO 2011

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

SKLÁDÁNÍ PANORAMAT Z FOTOGRAFIÍ

STITCHING PANORAMAS FROM PHOTOS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — LUKÁŠ POSPÍŠIL
AUTHOR

VEDOUCÍ PRÁCE — Ing. PAVEL SVOBODA
SUPERVISOR

BRNO 2011

Abstrakt

Hlavní náplní této práce je porovnání algoritmů pro výpočet homografie. Konkrétně jsou porovnávány algoritmy RANSAC, LO-RANSAC a PROSAC. Samotné skládání panoramat je bráno pouze jako prostředek pro porovnání. Ale i přesto jsou v této práci uvedeny základní algoritmy používané při skládání panoramat.

Abstract

The main purpose of this paper is a comparison of the algorithms for computing homography. It compares RANSAC, LO-RANSAC and PROSAC algoritms. Stiching panoramas is used just for their comparison, however the basic algoritms for stitching panoramas are mentioned.

Klíčová slova

skládání panoramat, panoramatická fotografie, detekce význačných bodů, SIFT, SURF, RANSAC, PROSAC, LO-RANSAC, homografie

Keywords

stitching panoramas, panoramic photo, interest point detection, SIFT, SURF, RANSAC, PROSAC, LO-RANSAC, homograhpy

Citace

Lukáš Pospíšil: Skládání panoramat z fotografií, bakalářská práce, Brno, FIT VUT v Brně, 2011

Skládání panoramat z fotografií

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Pavla Svobody

. . . . . .
Lukáš Pospíšil
17. května 2011

Poděkování

Rád bych poděkoval svému vedoucímu práce Ing. Pavlu Svobodovi za ochotu a pomoc při zpracování mé bakalářské práce.

© Lukáš Pospíšil, 2011.
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

Kapitola 1

Úvod

Při tvorbě panoramatických fotografií existuje několik postupů, jak je možné tyto snímky vytvořit. Nejjednodušší způsob je ořez běžné fotografie. Můžeme také použít speciální panoramatický fotoaparát, ale jeho pořízení je finačně nákladné. Nejrozšířenějším způsobem je tvorba panoramat složením z více fotografií. K tomuto postupu není potřeba žádný speciální přístroj a zároveň nedochází ke snížení rozlišení výsledného snímku, jak je tomu u tvorby pomocí ořezu fotografie. V dnešní době není problém získat software pro tvorbu panorama. Celá řada z nich je dokonce k dispozici zdarma (například Microsoft Image Composite Editor). Jak se čtenář sám přesvědčí, algoritmy, které se při tvorbě panoramat používají, nejsou složité.

V této práci jsem se rozhodl podrobněji prozkoumat část procesu skládání panoramat, která řeší problematiku výpočtu homografie a nejvíce ovlivňuje vzhled výsledného panorama. Pro řešení této problematiky se používá algoritmus RANSAC a jeho modifikace. Ve své práci bych chtěl provést srovnání několika těchto algoritmů. Z tohoto hlediska je v této práci skládání panoramat bráno jako prostředek pro porovnání algoritmů.

Práce je rozdělena do pěti kapitol (včetně úvodu). V druhé kapitole získá čtenář znalosti potřebné k porozumění této problematiky a rovněž se seznámí s algoritmy pro výpočet homografie, které jsem si vybral pro srovnání. Třetí kapitola je věnována návrhu programu. Zde se čtenář dozví návaznosti jednotlivých algoritmů, které byly představeny v druhé kapitole. Čtvrtá kapitola je věnována porovnání algoritmů pro výpočet homografie a jsou zde uvedeny i nedostatky programu, které byly během testování objeveny. Poslední kapitolou je závěr celé práce.

Program byl napsán v jazyce C/C++ s využitím knihovny OpenCV.

Kapitola 2

Teorie

Po přečtení této kapitoly čtenář získá teoreticky podklad k problematice skládání panoramat.

Pod pojmem panorama si každý představí minimálně dvě na sebe napojené fotografie. A právě výpočet napojení fotografií je náplní této práce. V počítačové grafice plní homografie roli napojení. Pro výpočet homografie potřebuje znát jaké body si odpovídají v jednotlivých fotografiích. Ale jelikož nejsme schopni určit tyto korespondence, je zde potřeba využít speciálních algoritmu, které nám určí jaké body si odpovídají. Z hlediska tvorby korespondencí (odpovídajících si bodů) je potřeba ve fotografiích zvolit body, které jsou nějakým způsobem významné. A tvorbu korespondencí nám usnadní pokud získáme popis vlastnosti těchto bodů, které můžeme porovnávat. A právě principy využívané v těchto situací jsou uvedeny v této kapitole.

2.1 SIFT

SIFT je zkratka z anglického Scale Invariant Feature Tranform. Na rozdíl od předchozích metod pro detekci významných bodů jsou body nalezené nezávislé na měřítku, rotaci a částečně na změnách osvětlení a nově zavádí popis významných bodů pomocí deskriptorů. Při tvorbě této kapitoly jsem vycházel z literatury [7] a [11].

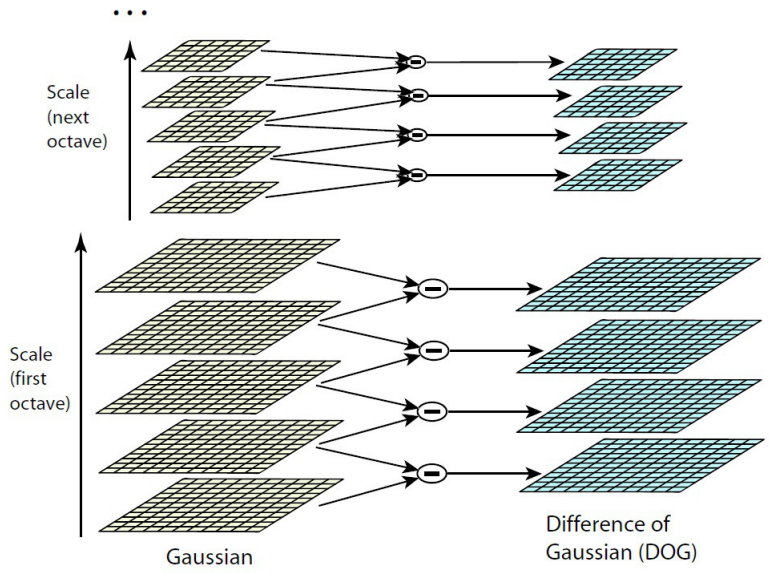
Princip celé metody je následující:

1. Vytvoření scale-space reprezentace a nalezení lokálních extrému
2. Odstranění nestabilních významných bodů
3. Přiřazení orientace
4. Vytvoření deskriptorů

2.1.1 Vytvoření scale-space reprezentace a nalezení lokálních extrémů

Scale-space je měřítkově nezávislá reprezentace původního obrázku. Scale-space vytvoříme tak, že z původního obrázku vytvoříme sadu stejných obrázků jen v různých měřítkách a provedeme konvoluci s Gaussovým jádrem (získáme rozmazané obrázky). Rozmazání obrázku v jednom měřítku pomocí Gaussova jádra je opakováno několikrát. Sada těchto rozmazaných obrázků (v jednom měřítku) se nazývá oktáva. Scale-space reprezentace je tvořena n oktávy, kde n je počet měřítek.

Když máme vytvořené oktávy, tak scale-space získáme odečtením dvou po sobě jdoucích vrstev (rozdíl Gausových funnkcí, znázorněno na obrázku 2.1).



Obrázek 2.1: Vytvoření space-scale pomocí rozdílu Gausových funkcí. Zdroj:[11]

Po vytvoření scale-space můžeme začít s hledáním kandidátů významných bodů. Kandidáti významných bodů leží v lokálních extrémech scale-space. Každý bod je otestován se svými osmi sousedy ve vrstvě a dalšími osmnácti body, s kterými sousedí ve vedlejších vrstvách. Kandidát tedy nemůže ležet na první ani poslední vrstvě oktávy a je testován ze svými dvaceti šesti sousedy. Bod je prohlášen za kandidáta, pokud je větší nebo menší než všichni jeho sousedi. U těchto bodů si zapamatujeme jeho lokaci (pozici a měřítko, v kterém se nachází).

2.1.2 Odstranění nestabilních významných bodů

Z kandidátů významných bodů potřebujeme odstranit nestabilní body, protože rozdíl Gaussovi funkce má silnou odezvu podél hran. Tyto špatně predikované body mají velkou hlavní křivost kolmo na hranu a malou podél hrany. Hlavní křivost lze vypočítat z Hessianovi matice $\mathcal{H}$.

$$\mathcal{H} = \begin{vmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{vmatrix}$$

Podle níže uvedené nerovnice 2.1 poznáme, zda bod zachovat, nebo ho ze skupiny významných bodů odstranit. Pokud nerovnice neplatí, bod je z množiny odstraněn. Hodnota r byla experimentálně stanovena na $r = 10$.

$$\frac{Tr(\mathcal{H})^2}{Det(\mathcal{H})} < \frac{(r+1)^2}{r} \tag{2.1}$$

kde $Tr(\mathcal{H}) = D_{xx} + D_{xy}$ a $Det(\mathcal{H}) = D_{xx}D_{xy} - (D_{xy}^2)$.

2.1.3 Přiřazení orientace

Aby bylo možné zajistit nezávislost na rotaci a měřítku obrázku, každému významnému bodu je přiřazena orientace a měřítko. Pro všechny body obrázku L je vypočítána velikost gradientu $m(x, y)$ a orientace $\theta(x, y)$.

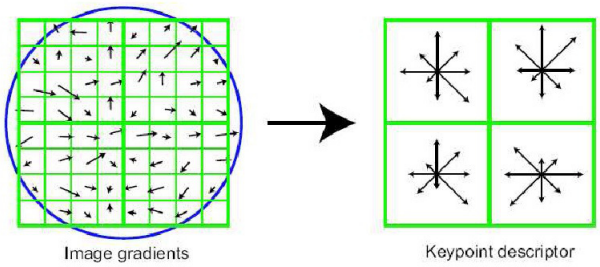
$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2}$$

$$\theta(x, y) = \tan^{-1} \frac{L(x, y+1) - L(x, y-1)}{L(x+1, y) - L(x-1, y)}$$

Z gradientů okolí významného bodu je vytvořen orientovaný histogram, který má 36 binů. Nejvyšší vrchol histogramu je prohlášen za orientaci významného bodu.

2.1.4 Vytvoření deskriptoru

Pro každý významný bod musí být vytvořen jedinečný deskriptor. Na Obrázku 2.2 je naznačeno, jak z gradientu okolí významného bodu je získán deskriptor. Vektory v malých čtvercích označují orientaci a velikost gradientu v daném bodě. Z těchto vektorů je pak vytvořen orientovaný histogram, který má 8 binů. Vektory jsou přiřazeny binu, který je nejblíže jejich úhlu. Pokud nejsou úhly binu a vektoru úplně stejné, je pomocí dalších výpočtů určena velikost vektoru ve směru binu.



Obrázek 2.2: Vytvoření deskriptoru. Zdroj:[11]

Deskriptor významného bodu bude vektor s n prvky (nebo bod v n dimenzích).

$$n = m * m * k$$

kde:

m je počet podčtverců v čtvercové oblasti deskriptoru významného bodu podél jedné strany

k je počet binů v orientovaném histogramu

Lowe a David při experimentech dosáhli nejlepších výsledků použitím 4×4 deskriptoru vytvořeného z okolí 16×16. Jako poslední krok se provádí na deskriptoru normalizace kvůli částečné nezávislosti na světelných změnách.

2.2 SURF

SURF je zkratka z anglického Speeded Up Robust Features. Tuto metodu v roce 2006 publikoval Herbert Bay [4]. Autoři při tvorbě této metody vycházeli z metody SIFT a snažili se o její zdokonalení. Výzkum této metody probíhal experimentálně. Možnosti pro zlepšení (úspory výpočetního času) viděli autoři hlavně ve fázi tvorby scale-space reprezentace a dále také v zjednodušení deskriptoru. Stejně jako SIFT, tak i SUFR je detektor nezávislý na měřítku, rotaci, změnách osvětlení a kontrastu. Při tvorbě této kapitoly jsem vycházel z literatury [4].

Princip metody je obdobný jako u metody SIFT:

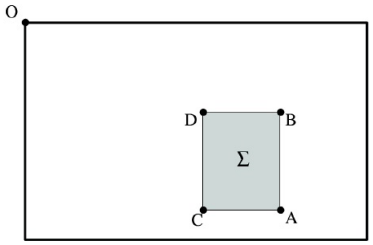
1. Vytvoření scale-space reprezentace, v které se následně hledají význačné body

2. Přiřazení orientace nalezeným bodům a jejich popis deskriptorem.

Tento detektor využívá aproximace Hessianovi matice, což vede k využití konceptu integrálních obrazců.

2.2.1 Integrální obrazce

Výhoda integrálních obrazců spočívá ve velmi rychlých výpočtech konvolucí s obdélníkovými filtry. Integrální obraz vytvoříme tak, že pro každý bod zdrojového obrázku vypočítáme jeho hodnotu integrálu od počátku po tento bod. Vzorec pro výpočet integrálu obdélníkové oblasti je $\sum = A - B - C + D$ (obrázek 2.3).



Obrázek 2.3: Použití integrálních obrazců. Zdroj: [4]

2.2.2 Detekce bodů Hessianovou matici

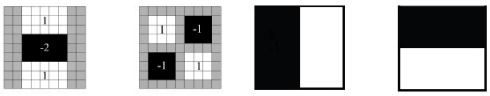
Detektor je založen na Hessianově matici, pro její dobrou výkonnost a přesnost. Detekujeme skvrnové struktury v místech, kde je determinant maximální.

Hessianova matice $\mathcal{H}(\mathbf{x}, \sigma)$ v bodě $\mathbf{x} = (x, y)$ a měřítku σ je definována :

$$\mathcal{H}(\mathbf{x}, \sigma) = \begin{vmatrix} L_{xx}(\mathbf{x}, \sigma) & L_{xy}(\mathbf{x}, \sigma) \\ L_{xy}(\mathbf{x}, \sigma) & L_{yy}(\mathbf{x}, \sigma) \end{vmatrix} ,$$

kde $L_{xx}(\mathbf{x}, \sigma)$ je konvoluce druhé Gausovi funkce $\frac{\partial^2}{\partial x^2} g(\sigma)$ s obrázkem I v bode $\mathbf{x}$ a podobně $L_{xy}(\mathbf{x}, \sigma)$ a $L_{yy}(\mathbf{x}, \sigma)$.

U detektoru SIFT se autoři snažili o co nejlepší aproximaci, ale experimentálně bylo zjištěno, že i při velmi nepřesné aproximaci (obdélníkovými filtry) bylo dosaženo kvalitních výsledků.



Obrázek 2.4: První dva filtry zleva jsou aproximace pro druhou derivaci Gaussovy funkce podle y (L_{yy}) a xy (L_{xy}), derivace podle x je otočený filtr pro derivaci podle y o 90 °. Další dva filtry jsou pro výpočet odezvy Haarových vlnek ve směru x a y. Zdroj: [4]

9×9 obdélníkové filtry na obrázku 2.4 (první dva) jsou aproximace Gausse s $\sigma = 1,2$ a reprezentuje nejmenší měřítko pro výpočet skvrnových map odezev. Budeme je označovat D_{xx}, D_{yy} a D_{xy}.

$$det(\mathcal{H}_{approx}) = D_{xx}D_{yy} - (wD_{xy})^2 \tag{2.2}$$

Relativní váha w výstupu filtrů je použita k vyvážení Hessianova determinantu. To je potřeba pro zachování energie mezi Gausovým jádrem a aproximovaným Gaussovym jádrem,

$$w = \frac{|L_{xy}(1,2)|_F |D_{yy}(9)|_F}{|L_{yy}(1,2)|_F |D_{xy}(9)|_F} = 0,912\ldots \simeq 0,9\ , \tag{2.3}$$

kde $|x|_F$ je Forbeniova forma.

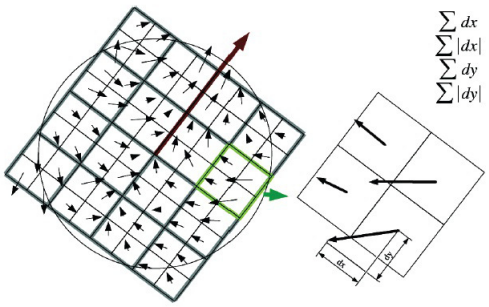
2.2.3 Scale-space reprezentace a lokalizace významných bodů

Díky použití obdélníkových filtrů (obrázek 2.4) a integrálních obrazců můžeme použít filtr přímo na originální obrázek, dokonce i paralelně. Proto je scale-space vytvořen pomocí zvětšujícího se filtru.

Scale-space je rozdělen do oktáv (stejně jako u metody SIFT). Oktáva reprezentuje sadu odezev filtrů získaných konvolucí stejného obrázku s filtrem, který je postupně zvětšován. Konstrukce scale-space začíná použitím filtru 9×9 a následně filtry 15×15, 21×21 a 27×27. Obdobně získáme další oktávy. Pro každou novou oktávu můžeme zvětšit krok zvětšení filtru na dvojnásobek (z 6 na 12 na 24 na 48). Významné body jsou pak stejně jako u metody SIFT nalezeny v lokálních maximech.

2.2.4 Orientace bodu a deskriptor

Abychom zajistili nezávislost na rotaci, potřebujeme stejně jako u metody SIFT získat orientaci významného bodu. Proto nejdříve v kruhovém okolí významného bodu o poloměru $6s$ vypočítáme odezvy Haarových vlnek (obrázek 2.4, pravá část) ve směru x (budeme značit d_x) a y (budeme značit d_y), kde s je měřítko, ve kterém byl bod nalezen. Pro výpočet může opět využít integrální obrazce.



Obrázek 2.5: Tvorba deskriptoru. Zdroj: [4]

Orientace je ustanovena jako suma všech orientací v okénku o velikosti 60 °. Toto okénko se posunuje až je ohodnoceno celé okolí významného bodu a největší orientace (vektor) je prohlášena za orientaci bodu.

Při konstrukci deskriptoru musíme nejdříve sestrojit čtvercové okno se středem v bodě, u kterého chceme zjistit deskriptor a natočit toto pole podle orientace (obrázek 2.5). Velikost okna je $20s$. Toto okno dále rozdělíme pravidelně na menší 4×4 čtvercové podoblasti. V těchto podoblastech vypočteme odezvu Haarových vlnek pro 5 bodů.

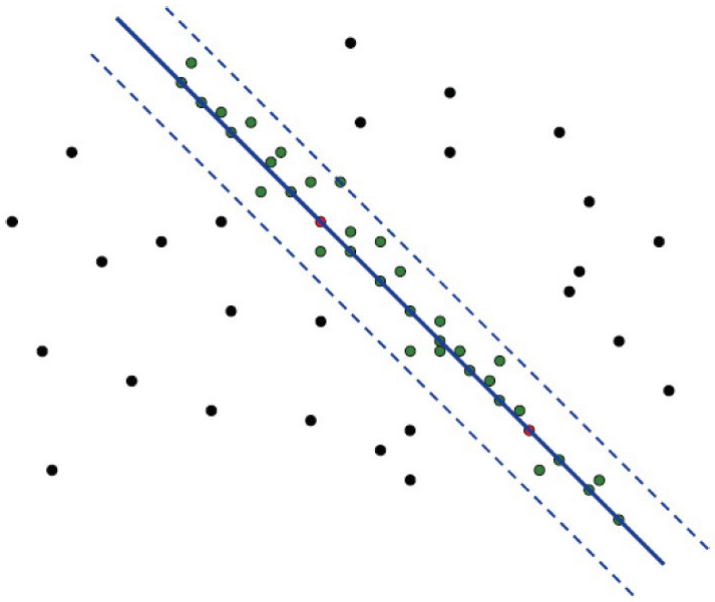
Deskriptor významného bodu je potom složen z šestnácti vektorů v (každé podoblasti), kde $v = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|)$. Velikost SURF deskriptoru je tedy 4*16 = 64.

Odezvy Haarových vlnek jsou nezávislé vůči změnám osvětlení a kontrastu, tudíž i metoda.

2.3 RANSAC

RANSAC je zkratka z anglického RANdom Sample Consensus [2]. Tuto metodu poprvé publikovali Dischler a Bolles v roce 1981[2]. Jedná se o velmi starou metodu, ale i přesto je velmi často využívána v počítačovém vidění (např. skládaní panoramatických fotek, detekce objektu, . . .). RANSAC je iterativní metoda pro výpočet parametrů matematického modelu ze zkoumaných dat. Při tvorbě této kapitoly jsem vycházel z literatury [2].

Základní předpoklad je takový, že vstupní data obsahují inliers (body, které popisují námi hledaný model) a outliers (body, které nepopisují námi hledaný model a znemožňují nám přímý výpočet parametrů modelu). Na základě náhodného výběru minimálního počtu bodů potřebného k sestavení hledaného modelu z vstupních dat získáváme ohodnocení, jak model sestavený z těchto bodů odpovídá hledanému modelu (jaká je kvalita těchto parametrů modelu). Nejlépe ohodnocený model během iterací je námi hledaný (v každé iteraci získáme ohodnocený model, který zahodíme pokud má horší ohodnocení než aktuálně nejlepší model, nebo jím nahradíme stávající). Počet iterací, kolikrát budeme muset vypočítat model z náhodných vzorků, abychom dosáhli určité pravděpodobnosti správnosti modelu, je definován rovnicí 2.6.



Obrázek 2.6: Detekce přímky algoritmem RANSAC. Zdroj: [10]

Příklad použití RANSACu naleznete na obrázku 2.6. Na tomto obrázku je algoritmus použit pro detekci přímky ve zdrojových datech. Obrázek je výsledkem nejlepší iterace algoritmu. Přímka byla nalezena na základě výpočtu z červených bodů. Zeleně zvýrazněné body jsou prohlášeny za inliers a černé body za outliers. Okolí přímky je ohraničeno modrou přerušovanou čárou a udává nám hranici pro určení inliers a outliers. Velikost této oblasti je volena dle řešené problematiky. Ohodnocení lze také rozšířit o součet vzdáleností inliers. V případě modelu, které jsou ohodnoceny stejným počtem inliers, máme jistotu, že bude zvolena lepší z nich.

Algoritmus lze vyjádřit pseudokódem [2]:

```
iterations := 0
bestModel := nil
bestConsensusSet := nil
bestError := infinity
while iterations < k
        maybeInliers := n // náhodně vybrané body z vstupních dat
        maybeModel := model // parametry k maybeInliers
        consensusSet := maybeInliers

        for všechny body z data not in maybeInliers
            if bod odpovídá maybeModelu s chybou menší než t
              přidej bod do consensusSet

        if počet prvků v consensusSet je > d
            // testujeme, jak dobrý je nový model
            betterModel := model // parametry odpovídají všem bodům v consensusSet
            thisError := a //měří, jak moc betterModel sedí s těmito body
            if thisError < bestError
              // našli jsme nový nejlépe ohodnocený model
              bestModel := betterModel
              bestConsensusSet := consensusSet
              bestError := thisError

        increment iterations

return bestModel, bestConsensusSet, bestError
```

Hodnoty parametrů t a d jsou získávány specificky podle aplikace a množiny dat. Parametr k (počet iterací) může být získán z teoretických výsledků. Ať p je pravděpodobnost, že algoritmus RANSAC v nějaké iteraci vybral pouze inliers ze vstupních dat, kde vybereme n bodů, z kterých vypočítáme parametry modelu. Nechť w je pravděpodobnost, že pokaždé je vybrán inlier, když je vybrán jeden bod.

$$w = \frac{pocet\ inliers\ v\ datech}{pocet\ bodu\ v\ datech} \tag{2.4}$$

Předpokládejme, že n bodů potřebných pro určení modelu je vybráno nezávisle. Pravděpodobnost w^n, že všechny body n jsou inliers a $1 - w^n$ pravděpodobnost, že alespoň jeden z bodů je outlier. Pak pravděpodobnost, že algoritmus nikdy nevybere n bodů, které jsou inliers je $1 - p$.

$$1 - p = (1 - w^n)^k \tag{2.5}$$

Počet iterací k potřebných k nalezení modelu je:

$$k\ = \frac{\log(1-p)}{\log(1-w^n)} \tag{2.6}$$

2.4 Lokálně optimalizovaný RANSAC (LO-RANSAC)

RANSAC je populární, protože je jednoduchý a pracuje dobře v praxi. Algoritmus tak funguje, protože o vstupních datech nic nepředpokládáme a nemusí být splněné žádné (nerealistické) podmínky pro úspěch algoritmu. Experimentálně však bylo zjištěno, že RANSAC běží déle, než by teoreticky měl. To je dáno předpokladem, který v praxi není pravdivý: předpokládá se model s parametry počítanými ze vzorku, který není kontaminovaný outliers. LO-RANSAC využívá skutečnosti, že modelová hypotéza z nekontaminovaného minimálního vzorku je téměř vždy dostatečně blízká přesnému řešení a použití lokálního optimalizačního kroku na určité modely vytváří algoritmus, který je téměř stejně efektivní, jak předpokládá teorie. Tento přístup zvyšuje počet nalezených inliers a urychluje RANSAC, protože umožňuje dřívější ukončení, ale také vrací kvalitnější modely.

Tato kapitola byla vytvořena překladem literatury [6].

2.4.1 Algoritmus

Struktura algoritmu je velmi jednoduchá. Opakovaně se náhodně vybírají vzorky ze vstupních dat a parametry modelu jsou vypočítány z těchto vzorků. Velikost náhodných vzorků je nejmenší možná pro určení parametrů modelu. Ve druhém kroku je určena kvalita parametrů modelu na základě kompletní skupiny hodnot (ohodnocení podle počtu bodů, které odpovídají modelu). Proces je ukončen, pokud je pravděpodobnost nalezení lepšího modelu malá, tzn. pravděpodobnost η chybějící sady inliers o velikosti I v rámci k vzorku spadne pod prahovou hodnotu

$$\eta = (1 - P_I)^k. \tag{2.7}$$

Princip algoritmu [6]:

Opakuj, dokud je pravděpodobnost nalezení lepšího řešení menší než prahová hodnota (rovnice 2.7).

1. Vyber si náhodný vzoreku minimálního počtu datových bodů Sm.
2. Odhadni parametry modelu, které odpovídají minimální skupině.
3. Spočítej počet inliers I_k (datových bodů s chybou menší než prahová hodnota θ).
4. Pokud nastalo nové maximum ($I_k > I_j$, pro všechna $j < k$), proveď **lokální optimalizaci**. Ulož nejlepší model.

P_I je pravděpodobnost, že nekontaminovaný vzorek o velikosti m je náhodně vybrán z N datových bodů

$$P_I = \frac{\binom{I}{m}}{\binom{N}{m}} = \prod_{j=0}^{m-1} \frac{I - j}{N - j} \approx \varepsilon^m \,, \tag{2.8}$$

kde ε je část inliers $\varepsilon = I/N$. Množství vzorků, které musí byt vybrány k zajištění η je

$$k \ = \log(\eta) / \log(1 - P_I). \tag{2.9}$$

Z rovnic 2.7 a 2.8 je patrné, že ukončovací kritérium založené na pravděpodobnosti η předpokládá, že výběr jednoho náhodného vzorku, který není kontaminován outliers, je následován nalezením celé sady inliers o velikosti I. Tento předpoklad není často platný, protože jsou inliers rozrušeny šumem. Protože RANSAC generuje hypotézy z minimálních sad, vliv šumu nelze zanedbat, a proto je nalezena odpovídající sada o velikosti menší než I. Z toho vyplývá navýšení

počtu vzorků, se kterými algoritmus pracuje. Navržená modifikace zvyšuje počet inliers nalezených blízko optimálnímu I. Toho je dosaženo použitím lokální optimalizace zatím nejlepších vzorků. Množství konzistentních datových bodů s modelem z náhodně vybraného vzorku může být považováno za náhodnou proměnnou s neznámou (nebo velmi komplikovanou) funkcí hustoty. Tato funkce hustoty je stejná pro všechny vzorky, takže pravděpodobnost, že k-tý vzorek bude zatím nejlepší je $1/k$. Průměrný počet dosažení prozatím nejlepšího vzorku v k vzorcích je

$$\sum_{1}^{k} \frac{1}{x} \leq \int_{1}^{k} \frac{1}{x}\, dx + 1 = \log k + 1.$$

2.4.2 Lokálně optimalizační metody

Tato podkapitola je doslovným překladem z literatury [6].

Výběr těchto metod je ovlivněn dvěma pozorováními, které jsou uvedeny dále.

1. **Standardní.** Běžná implementace RANSAC bez místní optimalizace.
2. **Jednoduchá.** Vezme všechny datové body s chybou menší než θ a použije lineární algoritmus pro odhad nových parametrů modelu.
3. **Iterativní.** Vezme všechny datové body s chybou menší než $K \cdot \theta$ a použije lineární algoritmus pro výpočet nových parametrů modelu. Sníží práh a iteruje, dokud je práh θ.
4. **Vnitřní RANSAC.** Je použita nová vzorkovací procedura. Vzorky jsou vybrány pouze z I_k datových bodů, které se shodují s předpokládaným modelem k-tého kroku RANSACu. Nové modely jsou verifikovány proti celé sadě datových bodů. Protože vzorkování běží na inliers datech, není důvod mít minimální velikost vzorku. Na druhé straně se velikost vzorku stanoví tak, aby se minimalizovala chyba odhadu parametrů modelu. V experimentech autorů metody je velikost vzorků nastavena na $\min(I_k/2, 14)$ pro epipolární geometrii a na $\min(I_k/2, 12)$ v případě odhadu homografie. V jejich experimentech je počet opakování nastaven na 10.
5. **Vnitřní RANSAC s iterací.** Tato metoda je stejná jako předchozí s tím rozdílem, že každý vzorek vnitřního RANSACu je zpracován metodou 3.

Lokální optimalizace jsou založeny na následujících pozorováních:

1. **Velikost vzorku**

   Čím méně informací (datových bodů) je použito k odhadu parametrů modelu za přítomnosti šumu, tím je model méně přesný. RANSAC vybírá minimální vzorky, protože každý další bod exponenciálně snižuje pravděpodobnost vzorku bez odlehlých bodů. Pravděpodobnost je ε^m, kde m je velikost vzorku (počet datových bodů, které jsou v něm zahrnuty).

2. **Iterativní schéma**

   Jak je známo z literatury o robustní statistice, pseudo-robustní algoritmy, které nejdříve odhadnou parametry modelu ze všech dat minimálně čtvercovou minimalizací, poté odeberou datové body s největší chybou (nebo zbytkové) a iterativně tento proces opakují. Toto však nevede k přesným odhadům. Může být snadno ukázáno, že jediný datový bod, který leží mimo (tzv. bod vlivu) dokáže úplně zničit odhadované parametry modelu. Bod vlivu má větší váhu než většina inliers s minimalizací metodou nejmenších čtverců. Tento algoritmus funguje dobře jen tehdy, pokud nejsou outliers převažující, takže převažující inliers mají větší vliv na metodu nejmenších čtverců.

V lokální optimalizační metodě 3 nejsou žádné body vlivu, protože každý datový bod má chybu menší než $K \cdot \theta$ příslušející vzorkovanému modelu.

2.5 PROSAC

PROSAC je zkratka z anglického PROgressive SAmple Consensus [2]. Tato metoda dosahuje velkých výpočetních úspor oproti RANSACu díky lineárnímu seřazení množiny $\mathcal{U}$ (množina předběžných přiřazení bodů, vstupní data). Prvky v množině $\mathcal{U}$ jsou seřazeny na základě funkce podobnosti $q(\cdot)$. Toto ohodnocení již známe z předchozího kroku, když jsme vytvářeli předběžné přiřazení bodů. Jako funkce $q(\cdot)$ se běžně používá korelace intenzit v okolí významného bodů, Mahalanobis vzdálenost neměnných deskriptorů, nebo podíl vzdáleností v SIFT prostoru prvního k druhému nejbližšímu sousedovi. Vzorky jsou náhodně vybírány z postupně zvětšující se množiny předběžných přiřazení. Zlepšení výkonu spočívá v předpokladu, že předběžné přiřazení s větší podobností jsou více pravděpodobněji inliers.

Tato kapitola byla vytvořena překladem literatury [5].

2.5.1 Algoritmus

Struktura algoritmu PROSAC je podobná algoritmu RANSAC. Opět jsou náhodně vybírány vzorky, ale narozdíl od algoritmu RANSAC nejsou vybírány ze všech dat, ale z podmnožiny tvořené nejlépe ohodnocenými prvky a jak již bylo řečeno, tato množina se postupně zvětšuje (prvky, které jsou pravděpodobněji inliers jsou vybrány dříve). Pro úspěch algoritmu musí byt vyřešeny dva problémy: růst funkce $n = g(t)$, který definuje množinu $\mathcal{U}_n$ tvořenou n nejlépe ohodnocenými přiřazeními, které jsou vybrány po t krocích. A ukončující kritérium, které zajišťuje podobnost k algoritmu RANSAC v optimalitě získaného řešení. Vzorky $\mathcal{M}$ jsou vybírány z podmnožiny datových bodů $\mathcal{M} \subset \mathcal{U}_N, |\mathcal{M}| = m$, kde m je velikost vzorku. Jako kvalitu vzorku pak označujeme nejmenší kvalitu z bodů ve vzorku $q(\mathcal{M}) = \min q(u_i), \quad u_i \in \mathcal{M}$.

Princip algoritmu [5]:

$t := 0,\ n := m,\ n^* := N$

Opakuj dokud je nalezeno řešení vyhovující rovnicím 2.16 a 2.19.

1. **Volba hypotézy generování množiny**

   $t := t + 1$

   if $(t \geq T'_n)$ & $(n < n^*)$ then $n := n + 1$ (viz. rovnice 2.12)

2. **Semi-náhodný vzorek $\mathcal{M}_t$ o velikosti m**

   if $T'_n < t$ then

   Vzorek obsahuje $m - 1$ bodů vybraných z $\mathcal{U}_{n-1}$ náhodně a u_n

   else

   Výběr m bodů z $\mathcal{U}_n$ náhodně

3. **Odhad parametrů modelu**

   Vypočti parametry modelu p_t z vzorku $\mathcal{M}_t$

4. **Verifikace modelu**

   Najít podporu modelu s parametry p_t

   Pokud je to možné, zvolit ukončovací delku n^*

2.5.2 Růst funkce a výběr vzorku

Vycházíme z předpokladu, že pravděpodobnost $P\{u_i\}$ (přiřazení u_i je správné) a funkce podobnosti $q(u_j)$ mají určitou spojitost. Konkrétně předpokládáme monotónnost

$$i < j \Rightarrow q(\mathcal{M}_{(i)}) \geq q(\mathcal{M}_{(j)}). \tag{2.10}$$

Strategie výběru vzorku. Obrazový standard RANSAC vybírá T_N vzorků o velikosti m z N datových bodů. $\{\mathcal{M}_i\}_{i=1}^{T_N}$ označuje sled vzorků $\{\mathcal{M}_i\} \subset \mathcal{U}_N$, které jsou rovnoměrně vybrány metodou RANSAC a $\{\mathcal{M}_{(i)}\}_{i=1}^{T_N}$ je stejný sled vzorků seřazených sestupně podle kvality vzorku, $i < j \Rightarrow q(\mathcal{M}_{(i)}) \geq q(\mathcal{M}_{(j)})$. Pokud jsou tedy vzorky brány v pořadí $\mathcal{M}_{(i)}$, tak jsou nejdříve vybrány vzorky, které jsou pravděpodobně nekontaminované. Po T_N vzorcích jsou vybrány všechny vzorky jako u metody RANSAC $\{\mathcal{M}_i\}_{i=1}^{T_N}$.

Nechť T_n je průměrné číslo vzorků z $\{\mathcal{M}_i\}_{i=1}^{T_N}$, které obsahuje datové body pouze z U_n

$$T_n = T_N \frac{\binom{n}{m}}{\binom{N}{m}} = T_N \prod_{i=1}^{m-1} \frac{n-i}{N-i}, \text{ pak}$$

$$\frac{T_{n+1}}{T_n} = \frac{T_N}{T_N} \prod_{i=1}^{m-1} \frac{n+1-i}{N-i} \prod_{i=1}^{m-1} \frac{N-i}{n-i} = \frac{n+1}{n+1-m}.$$

Pak opakující se vztah pro T_{n+1} je

$$T_{n+1} = \frac{n+1}{n+1-m} T_n. \tag{2.11}$$

Máme T_n vzorků, které obsahují pouze datové body z $\mathcal{U}_n$ a T_{n+1} vzorků, které obsahují pouze datové body z $\mathcal{U}_{n+1}$. Protože $\mathcal{U}_{n+1} = \mathcal{U}_n \cup \{u_{n+1}\}$, existuje $T_{n+1} - T_n$ vzorků, které se skládají z datového bodu u_{n+1} a $m-1$ datových bodů vybraných z $\mathcal{U}_n$. Takže procedura, která pro $n = m \ldots N$ vybírá $T_{n+1} - T_n$ vzorků, které se skládají z datového bodu u_{n+1} a $m-1$ náhodně vybraných datových bodů z $\mathcal{U}_n$. Tato procedura efektivně generuje vzorky $\mathcal{M}_{(i)}$.

Hodnoty T_n nemusí být typu integer, definujeme $T'_m = 1$ a

$$T'_{n+1} = T'_n + [T_{n+1} - T_n]. \tag{2.12}$$

Růst funkce je pak definován jako

$$g(t) = \min\{n : T'_n \geq t\}. \tag{2.13}$$

2.5.3 Ukončovací kritérium

Algoritmus PROSAC se ukončí, pokud počet inliers I_{n^*} v množině $\mathcal{U}_{n^*}$ splňuje následující podmínky [5]:

- **nenáhodnost** - pravděpodobnost, že I_{n^*} z n^* datových bodů jsou pravděpodobné inliers jakéhokoli chybného modelu, je menší než Ψ (běžně nastaveno na 5%)
- **maximálnost** - pravděpodobnost, že existuje řešení, které má více než I_{n^*} inliers v množině $\mathcal{U}_{n^*}$ a nebylo nalezeno po k vzorcích je menší než η_0 (běžně nastaveno na 5%)

Ze všech těchto řešení vybereme to, které ukončení algoritmu způsobí jako první.

Podmínka **nenáhodnosti** zabrání algoritmu vybrat řešení na základě outliers, které by mu mohly odpovídat. Omezení je běžně kontrolováno dodatečně. Rozdělení kardinalit množin náhodných inliers je binomické

$$P_n^R(i) = \beta^{i-m}(1-\beta)^{n-i+m}\binom{n-m}{i-m}, \tag{2.14}$$

kde β je pravděpodobnost, že nesprávný model vypočítaný z náhodného vzorku, který obsahuje outliers, je podporován přiřazením, které není zahrnuto ve vzorku.

β nastavujeme pesimisticky na základě geometrie a pokud je potřeba, může být odhad β upřesněn během výběru vzorků metody PROSAC.

Pro každé n je minimální počet inliers $I_n^{\min}$ vypočítán, takže pravděpodobnost, že je velikost takové podpory náhodná, je menší než Ψ

$$I_n^{\min} = \min\{j : \sum_{i=j}^{n} P_n^R(i) < \Psi\}. \tag{2.15}$$

Nenáhodné řešení nalezené v množině $\mathcal{U}_n$ musí splňovat

$$I_{n^*} \geq I_{n^*}^{\min}. \tag{2.16}$$

Podmínka **maximálnosti** definuje minimální počet nutných vzorků pro ověření správnosti řešení a je to jediná ukončující podmínka algoritmu RANSAC.

Pro hypotetickou generátorovou množinu $\mathcal{U}_n$ má pravděpodobnost P_{I_n}, že nekontaminovaný vzorek o velikosti m je náhodně zvolen z množiny $\mathcal{U}_n$ z n přiřazení, hodnotu

$$P_{I_n} = \frac{\binom{I_n}{m}}{\binom{n}{m}} = \prod_{j=0}^{m-1}\frac{I_n - j}{n-j} \approx \varepsilon_n^m, \tag{2.17}$$

kde I_n je počet inliers v $\mathcal{U}_n$ a $\varepsilon_n = I_n/n$ je část inliers. Pravděpodobnost η, že nám chybí množina inliers o velikosti I_n z množiny $\mathcal{U}_n$ po k vzorcích, pokud $g(k) \leq n$, je rovna

$$\eta = (1 - P_{I_n})^k. \tag{2.18}$$

Počet vzorků, který musí být vybrán, aby bylo zajištěno, že pravděpodobnost η klesne pod zadaný práh η_0 je

$$k_{n^*}(\eta_0) \geq \log(\eta_0)/\log(1 - P_{I_n^*}). \tag{2.19}$$

Ukončovací délka n^* je zvolena tak, aby minimalizovala $k_{n^*}(\eta_0)$, které závisí na $I_{n^*} \geq I_{n^*}^{\min}$.

2.6 Geometrické transformace

Geometrické transformace jsou jedním z nejdůležitějších a nejpoužívanějších nástrojů moderní počítačové grafiky. Pojmu geometrické transformace rozumíme jako nástroji pro změnu pozice bodu. Při tvorbě této kapitoly jsem vycházel z literatury [9].

Při používaní transformací se využívá homogenních souřadnic. Homogenní souřadnice rozšiřují popis bodu o jednu dimenzi a to z toho důvodu, aby bylo možné sjednotit operace posunu, otočení, změny měřítka a zkosení. Homogenní souřadnice bodu ve 2D tedy budou $A(x, y, w)$, kde w nazýváme vahou bodu a v případě lineárních transformací je většinou $w = 1$. Lineární transformace jsou všechny základní transformace (níže uvedené).

1. **Posunutí** - posune bod ve směru os x a y o d_x a d_y. Nové souřadnice pak získáme podle vztahu:
$$x' = x + d_x, \quad y' = y + d_y$$
a transformační matice má tvar:
$$T = \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ d_x & d_y & 1 \end{vmatrix}$$

2. **Otáčení** - otočí bod o úhel α se středem otáčení v počátku souřadného systému. Nové souřadnice pak získáme podle vztahu:
$$x' = x \cdot \cos\alpha - y \cdot \sin\alpha, \quad y' = x \cdot \sin\alpha + y \cdot \cos\alpha$$
a transformační matice má tvar:
$$T = \begin{vmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

3. **Změna měřítka** - změní měřítka bodu ve směru os x a y podle faktorů změny měřítka S_x a S_y. Pro faktor změny měřítka S platí: $S > 1$ zvětšení, $0 < S < 1$ zmenšení a $S < 0$ převrácení (zrcadlení).
Nové souřadnice pak získáme podle vztahu:
$$x' = x \cdot S_x, \quad y' = y \cdot S_y$$
a transformační matice má tvar:
$$T = \begin{vmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

4. **Zkosení** - zkosí bod ve směru os x a y podle faktorů zkosení S_{hx} a S_{hy}. Nové souřadnice pak získáme podle vztahu:
$$x' = x + S_{hx} \cdot y, \quad y' = y + S_{hy} \cdot x$$
a transformační matice má tvar:
$$T = \begin{vmatrix} 1 & S_{hx} & 0 \\ S_{hy} & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

V praxi většinou nestačí použití jedné transformace, ale potřebujeme aplikovat sadu transformací. Proto postupným složením (násobením) jednotlivých základních transformací dostaneme jednu výslednou transformaci. Při skládání (násobení) transformačních matic je důležité dodržet jejich pořadí. Přehození pořadí by vedlo k získaní jiné výsledné transformace než potřebujeme.

2.7 Homografie

Tato kapitola byla převzata z literatury [8].

Homografie je mapování bodů z jedno obrazu do druhého (vyjadřuje transformaci mezi obrazy). V 2D je homografie definována jako matice H o rozměrech 3×3, která spojuje body p v jednom obrazu s body p' v druhém obraze.

$$wp' = Hp$$

Kde w je parametr měřítka. Homografie má pouze 8 stupňů volnosti, takže může být získána pomocí čtyř korespondencí mezi dvěma obrazy. Jestliže $(x, y, 1)$ a $(x', y', 1)$ je pár odpovídajících si bodů, pak získáme:

$$\begin{vmatrix} wx' \\ wy' \\ w \end{vmatrix} = \begin{vmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{vmatrix} \begin{vmatrix} x \\ y \\ 1 \end{vmatrix}$$

kde eliminací w získáme dvě rovnice.

$$h_{11}x + h_{12}y + h_{13} - h_{31}xx' - h_{h32}yx' - x' = 0$$

$$h_{21}x + h_{22}y + h_{23} - h_{31}xy' - h_{h32}yy' - y' = 0$$

Jelikož lze každé korespondence popsat dvěmi rovnicemi, tak ze čtyř korespondencí získáme lineární systém o osmi rovnicích s osmi proměnnými. Tento lineární systém pak můžeme zapsat jako $Ah = 0$, kde předpokládáme n korespondenci, A je matice $2n \times 9$ získaná:

$$A = \begin{vmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x'_1 & -y_1x'_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1x'_1 & -y_1y'_1 & -y'_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2x'_2 & -y_2x'_2 & -x'_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2x'_2 & -y_2y'_2 & -y'_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & 1 & 0 & 0 & 0 & -x_nx'_n & -y_nx'_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -x_nx'_n & -y_ny'_n & -y'_n \end{vmatrix}$$

a h je vektor 9×1 osahující všechny prvky homografie H. Nyní můžeme jednoduše získat homografii H použitím SVD (Singular Value Decomposition, více informací o této problematice naleznete v literatuře [3]) matice A.

Kapitola 3

Skládání panoramat

Jak bylo naznačeno v úvodu, program je zaměřen na porovnání algoritmů pro výpočet homografií. Z tohoto důvodu je program značně limitován (viz uživatelský manuál). Panoramata jsou skládána pouze ze dvou zdrojových fotografií a výsledné panorama je vytvořeno pouhým nakopírováním fotografií přes sebe. V programech přímo pro skládání panoramat se tato problematika řeší zcela jiným způsobem, ale pro potřeby této práce tento přístup zcela postačuje. Po samostatném spojení jsou profesionálními programy následně provedeny další úpravy panorama (např. blending, mapování na válec, ...).

Program je napsán v jazyce C/C++ s využitím knihovny OpenCV. Program není implementován objektově, pouze využívá některé funkce ze základních tříd C++. Knihovna OpenCV je zaměřená na počítačové vidění a zpracování obrazu. Tudíž velmi usnadňuje práci s fotografiemi a některé algoritmy jsou již její součástí. Program byl testován na verzi OpenCV 2.1.

3.1 Návrh programu

Základním předpokladem při skládání panoramat je, že zdrojové fotografie mají část snímané scény společnou (z části se překrývají). A z této myšlenky vychází i program. Ve zdrojových fotografiích se nejdříve nadetekuje několik bodů a z těchto bodů hledáme takové, které jsou pro obě fotografie společné. Když nalezneme tyto společné body, provést spojení fotografií je již velmi jednoduché. Tyto body se dají na sebe (transformují) a získáme hledané panorama. Z této základní myšlenky odvodíme jednotlivé kroky, které musí program postupně provést:

1. Načíst zdrojové fotografie
2. Nalézt významné body ve zdrojových fotografiích
3. Určit korespondence mezi fotografiemi
4. Vypočítat homografii mezi fotografiemi
5. Vypočítat velikost panorama a pozici jednotlivých fotografií
6. Nakopírovat fotografie do panorama
7. Provést úpravy panorama (tento krok v práci neřeším)

Program pro automatické skládání panorama by měl mít uvedenou skladbu. Výsledné panorama je ovlivněno všemi těmito kroky. Největší vliv na výsledek mají body 2 a 4.

Bez dobře nadetekovaných bodů nemůže program v dalších krocích podávat dobré výsledky. Tento fakt potvrzuje i neustálý vývoj v oblasti detekce bodů. Prvotní algoritmy jako Moravcův rohový detektor, byly vylepšeny a nahrazeny jinými. V dnešní době jsou velmi používané algoritmy SIFT a SURF. Je důležité dodat, že algoritmy pro detekci významných bodů jsou v oblasti počítačového vidění velmi důležité, a proto lze v budoucnosti očekávat další nové algoritmy.

Druhým velmi důležitým krokem je výpočet homografie. Zde se využívá algoritmu RANSAC a jeho modifikací. Bez těchto algoritmů by automatické generátory panoramat jen těžko fungovaly. V této práci byl čtenář seznámen s třemi algoritmy, ale existuje velké množství dalších (např.[6] MLESAC, R-RANSAC, NAPSAC, . . .). Tyto algoritmy dokážou eliminovat i nedostatk při fázi určování korespondencí za cenu delšího výpočetního času.

3.2 Načtení fotografií

Fotografie jsou načteny pomoci funkce `cvLoadImage` z knihovny OpenCV. Obrázky jsou načteny dvakrát. Jednou jsou načteny barevně (tři barevné kanály) a jsou použity jako zdrojové snímky do panorama. Podruhé jsou načteny ve stupních šedi, protože při detekci významných bodů využívám funkci z knihovny OpenCV, která to vyžaduje. Funkce `cvLoadImage` podporuje tyto formáty obrázků: BMP, DIB, JPEG, JPG, JPE, PNG, PBM, PGM, PPM, SR, RAS, TIFF, TIF. Program tedy podporuje pouze zdrojové fotografie v těchto formátech.

3.3 Detekce významných bodů

Jako detektor významných bodů jsem zvolil robustní algoritmus SURF. Z kapitoly 2.2 víme, že tento algoritmus nezjistí pouze pozice významných bodů, ale také nám tyto body popisuje pomocí deskriptorů, což nám velmi usnadňuje práci ve fázi hledání korespondencí. U algoritmu SURF jsem využil jeho implementaci v knihovně OpenCV, funkce `cvExtractSURF`. Jak bylo zmíněno v kapitole 3.2, tato funkce vyžaduje fotografii, ve které chceme nadetekovat významné body, ve stupnici šedi. Tento fakt nám ovšem nevadí, protože významné body budou mít v barevném obrázku pozici úplně stejnou. Příklad nadetekovaných významných bodů najdeme na obrázku 3.1.

Obrázek 3.1: Detekce významných bodů pomocí algoritmu SURF (1287 a 1044 bodů)

3.4 Určení korespondencí mezi fotografiemi

Když získáme nadetekované významné body a jejich popis pomocí deskriptorů, máme vše potřebné k určení korespondencí mezi fotografiemi. Je důležité dodat, že naším úkolem v tomto kroku není určit pouze správné korespondence (což by bylo nemožné), ale pouze určit předpoklad jaké body si budou odpovídat. Získáme množinu korespondencí, mezi kterými budou i nesprávné korespondence, protože se jedná pouze o odhad.

Ve svém programu mám problematiku korespondencí velmi zjednodušenou díky faktu, že program vyžaduje zadat zdrojové fotografie jak na sebe budou navazovat, přičemž je panorama složeno pouze ze dvou fotografií. Odpadá tedy problém určení, které fotografie na sebe navazují. Výsledné panorama budu tvořit namapováním pravé fotografie do levé. Množina korespondencí se proto dá zmenšit pouze na korespondence bodů z pravé fotografie do bodů v levé fotografii. Samotný princip hledání korespondencí je takový, že vezmu všechny nadetekované významné body v pravé fotografie a na základě porovnání deskriptorů určím, který bod z levé fotografie je mu nejvíce podobný. Pro zmenšení množiny (následné urychlení algoritmu pro výpočet homografie) jsou vyřazeny korespondence, které mají podobnost deskriptoru menší než předdefinovaná hodnota.

Funkce `findPairs`, `naiveNearestNeighbor` a `compareSURFDescriptors` obsahují implementaci této části. Tyto funkce jsem převzal z ukázkových příkladů knihovny OpenCV (`find_obj.cpp`). Příklad množiny předpokládaných korespondencí můžeme vidět na obrázku 3.2 a 3.3.

3.5 Výpočet homografie

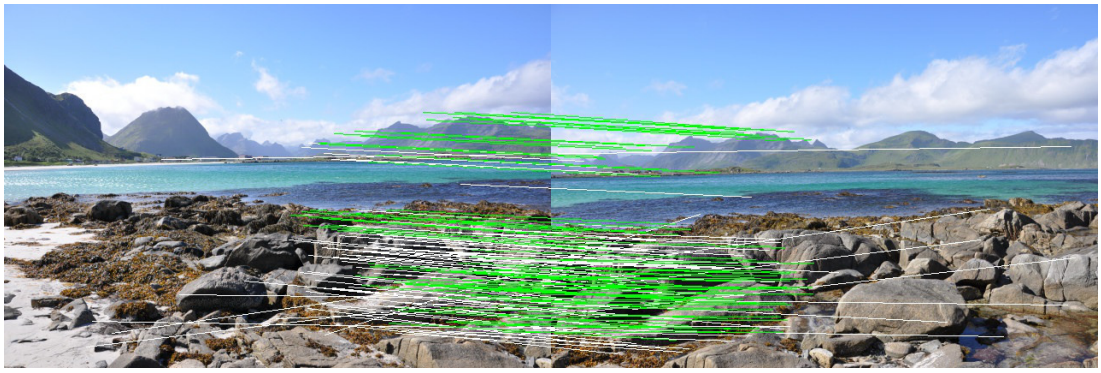
Výpočet homografie je jedna z nejdůležitější částí celého programu. Homografie má největší dopad na vzhled výsledného panorama. Samozřejmě tyto algoritmy vyžadují dobrá vstupní data (nadetekované významné body, z kterých jsou následně určeny předpokládané korespondence). Homografie nám udává, jak jsou fotky mezi sebou propojeny a jak na sebe navazují. Pokud fotky nejsou správně proloženy, nepomůže nám ani nejlepší program na úpravu fotografií. Finální úpravy panorama se dají udělat dodatečně v libovolném programu určenému na úpravu fotek.

U výpočtu homografie se setkáváme se zásadním problémem, protože ze vstupních dat nevíme, které korespondence jsou správné (popisují námi hledanou homografii), a které jsou špatné. Na základě porovnání deskriptorů bodů nemůžeme jednoznačně určit, zda jsou tyto body shodné, nebo jen velmi podobné (i přes sofistikovaný výpočet deskriptoru u algoritmu SURF to nelze na sto procent určit). Tento problém se řeší pomocí algoritmu RANSAC a jeho modifikací.

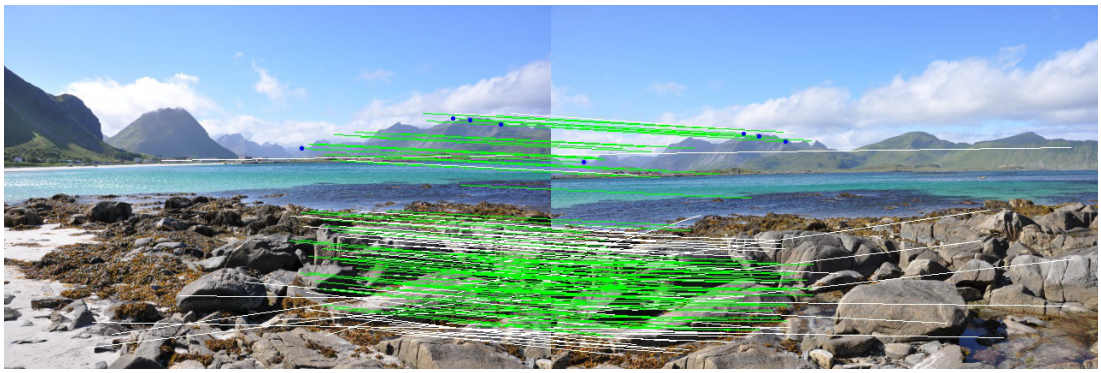
Ve svém programu jsem implementoval tři takovéto algoritmy: RANSAC, lokálně optimalizovaný RANSAC a PROSAC. Pro připomenutí si dovolím shrnout, jak tyto algoritmy pracují.

Na základě náhodného výběru minimálního počtu korespondencí (z kapitoly 2.7 víme, že se jedná o 4 korespondence) se vypočte homografie. Tento krok je několikrát opakován a výsledkem je nejlépe ohodnocená homografie.

Homografie ohodnocuji podle počtu inliers, které jim odpovídají a celkové vzdálenosti těchto inliers. Ohodnocení pouze na základě inliers není dostačující. Může nastat situace, že bude nalezeno stejně inliers jako v dosavadní nejlépe ohodnocené homografii. V tomto případě je porovnávána celková vzdálenost všech inliers (jako inliers jsou prohlášeny všechny korespondence, u kterých je euklidovská vzdálenost transformovaného bodu a odpovídajícího bodu menší než přednastavená hodnota). Jako generátor náhodných čísel využívám pseudonáhodný generátor, který je inicializován podle času, aby nedocházelo k výběru stejných vzorků. U náhodně vybraných korespondencí je důležité provádět kontrolu, aby náhodně nebyla vybrána žádná korespondence



Obrázek 3.2: Předpokládané korespondence. Zeleně jsou zvýrazněny inliers korespondence získané algoritmem LO-RANSAC. Algoritmus RANSAC pracuje v tomto ohledu stejně a určil by stejné inliers (74 inliers).



Obrázek 3.3: Předpokládané korespondence. Zeleně jsou zvýrazněny inliers korespondence získané algoritmem PROSAC (94 inliers). Body, z kterých byla vypočítána homografie, jsou zvýrazněny modře.

více jak jednou v rámci jednoho vzorku. Pro samotný výpočet homografie využívám funkci `cvGetPerspectiveTransform` z knihovny OpenCV.

U lokálně optimalizovaného RANSACu jsem zvolil jednoduchou lokální optimalizaci. Pro výpočet homografie z více jak čtyř korespondencí využívám funkci `cvFindHomography` z knihovny OpenCV. Tento algoritmus se bude z hlediska provedených iterací a nalezených inliers chovat stejně jako RANSAC, ale díky přepočtu homografie přes všechny nalezené inliers, bude výsledná homografie přesnější.

U výpočtu homografie by mohl nastat ještě další problém. Pokud by v náhodně vybraném vzorku ležely tři body na jedné přímce. Ale z následujících dvou důvodů jsem tento problém neřešil:

1. Ohodnocení homografie vypočtené z tohoto vzorku bude horší než ze správně zvoleného vzorku

2. U algoritmu PROSAC by mohlo dojít k zacyklení v případě, kdy množinu pro náhodný výběr tvoří pouze čtyři prvky a tři prvky by ležely na přímce

3.6 Výsledné panorama

Poslední část, která zbývá před dokončením, je výpočet rozměrů panorama a samotné spojení fotek. Jelikož spolu tyto dvě časti úzce souvisí, spojil jsem je do jedné kapitoly.

Výpočet rozměru panorama nám značně usnadňuje fakt, že jsou fotografie zadány postupně a transformujeme pravou fotografii do levé (zmíněno již v kapitole 3.5). Z předchozího kroku známe homografii a pouze zjistíme pozice, kam se transformují rohy pravé fotografie. Šířka je rovna větší hodnotě x z pravého horního a pravého dolního rohu. Výška je určena obdobně. Vypočte se y souřadnice horních a dolních rohů pravé fotografie. Pozice horních rohů nás zajímají i z dalšího důvodu. Pokud je souřadnice y horních rohů záporná, je před nakopírováním fotografií do panorama potřeba provést transformaci pro posun po ose y (stejně tak i pravou fotografii). Tento postup je implementován ve funkcích `sizeFinal` a `getMoveM`. Pro násobení matic je využita funkce `cvMatMul` z knihovny OpenCV.

Výsledné panorama je vytvořeno aplikováním homografie na pravý obrázek (pomocí funkce `cvWarpPerspective` z knihovny OpenCV). Pokud je potřeba provést i transformaci po ose y, výslednou transformační matici získáme vynásobením matice pro posun po ose y a homografie. Levý obrázek je přímo nakopírován od počátku panorama. Pokud byla aplikována transformace po ose y, nakopíruje se obrázek s příslušným posunem.

Obrázek 3.4: Panorama, u kterého pro výpočet homografie byl využit algoritmus RANSAC

Obrázek 3.5: Panorama, u kterého pro výpočet homografie byl využit algoritmus LO-RANSAC

Obrázek 3.6: Panorama, u kterého pro výpočet homografie byl využit algoritmus PROSAC

Kapitola 4

Výsledky experimentů

4.1 Porovnání algoritmů

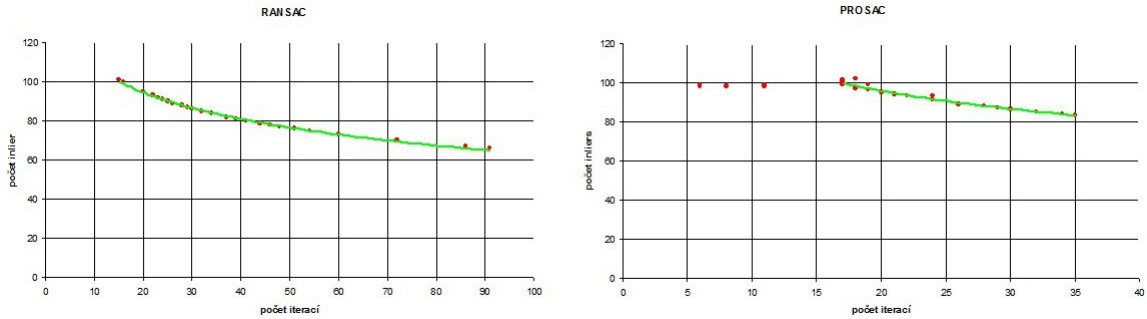
Algoritmy RANSAC, LO-RANSAC a PROSAC budu porovnávat z hlediska počtu nalezených inliers, iterací potřebných k určení homografie, času potřebných k určení homografie a kvality homgrafie (jak kvalitně jsou na sebe fotografie napojeny).

Dříve než se dostaneme ke konkrétním hodnotám, provedu rozbor, jaké hodnoty můžeme očekávat. RANSAC a LO-RANSAC s jednoduchou lokální optimalizací (dále pouze LO-RANSAC) by měly podávat stejné výsledky v počtu iterací algoritmu a nalezených inliers. Z hlediska porovnávaní těchto veličin jsou algoritmy totožné. Hodnoty nemohou být přesně stejné, protože se jedná o algoritmy založené na náhodném výběru, ale tyto výsledné hodnoty by se měly lišit pouze minimálně. Výpočetní čas algoritmu LO-RANSAC by měl být delší než u algoritmu RANSAC (provádí se navíc lokální optimalizace). U porovnání RANSACu a PROSACu z hlediska časové náročnosti není možné provést žádné predikce. Algoritmus RANSAC je méně složitý (kratší výpočetní čas), ale algoritmus PROSAC by měl provádět méně výpočetních iterací. Přednosti LO-RANSACu je jednoznačně kvalita výsledného napojení fotografií. Zde očekávám lepší výsledky než u zbývajících dvou algoritmů. PROSAC by měl dosahovat nejmenších čísel v počtu iterací. Z hlediska počtu nalezených inliers má lepší teoretické předpoklady algoritmus PROSAC. Zde ovšem může nastat problém vysokého ohodnocení podobných bodů, které si ovšem neodpovídají a může dojít k horší homografii vlivem této korespondence (tento bod může být náhodně vybrán několikrát, u PROSACu se množina předběžných korespondencí rozrůstá postupně). Díky využití kvalitního detektoru významných bodů SURF, který popisuje body velmi kvalitně, je tato možnost menší než při využití starších detektorů. PROSAC by mohl i z hlediska počtů inliers podávat nejlepší výsledky.

Protože jsou algoritmy založené na náhodném výběru, je třeba testování provést několikrát. Tímto krokem částečně eliminujeme extrémní výchylky. Hodnoty v tabulce 4.1 jsou průměrná čísla z padesáti provedených experimentů. Testy byly prováděny se zdrojovými fotografiemi z obrázku 3.1. Grafické znázornění experimentů nalezneme v grafech na obrázku 4.1 (graf pro LO-RANSAC by vypadal téměř stejně jako u RANSACu, protože z hlediska porovnávaných veličin se chová stejně).

algoritmus	**inliers**	**iterací**	**výpočetní čas**
RANSAC	83,88	37,16	4,44735 ms
LO-RANSAC	82,82	35,56	7,90502 ms
PROSAC	93,90	18,02	3,30688 ms

Tabulka 4.1: Výsledky experimentů.



Obrázek 4.1: Grafické znázornění experimentů

Obrázek 4.1 nám potvrzuje teoretické tvrzení, že PROSAC v nejhorším případě dosahuje stejných výsledků jako RANSAC (podobnost proložených křivek). Body, kterými křivka není proložena u algoritmu PROSAC, jsou dokumentací lepších výsledků než dosahuje RANSAC (způsobeno ukončujícím kritériem na základě kvality modelu). Během testování tato situace nastala sedmkrát (body v grafu se překrývají, a proto se zdá, že tato situace nastala pouze třikrát).

U porovnání algoritmů z hlediska kvality homografie nelze měřit žádnou veličinu na základě které bychom rozhodli o lepším výsledku. Toto vyhodnocení bylo provedeno na základě pohledu na panorama a sledováním, který algoritmus podával nejstabilnější výsledky. Jako nejlepší algoritmus v tomto směru jsem vyhodnotil LO-RANSAC. Příklad, jak se liší panorama získané pomocí algoritmu RANSAC (PROSAC, by ze stejných zdrojových bodů provedl stejné spojení fotografií) a algoritmu LO-RANSAC nalezneme na obrázcích 4.2 a 4.3 (v popředí u obrázku 4.2 můžeme vidět nepřesné napojení, které LO-RANSAC odstranil). Při tvorbě těchto panoramat byly algoritmy prohlášeny stejné korespondence jako inliers. Algoritmus PROSAC podával stabilnější výsledky než algoritmus RANSAC. Proto by mohl algoritmus PROSAC s jednoduchou lokální optimalizací podávat velmi zajímavé výsledky.

Obrázek 4.2: Panorama vytvořené pomocí algoritmu RANSAC

Shrnuti algoritmu se nachází v kapitole 4.4.

Obrázek 4.3: Panorama vytvořené pomocí algoritmu LO-RANSAC

4.2 Ukončující kriterium algoritmu RANSAC

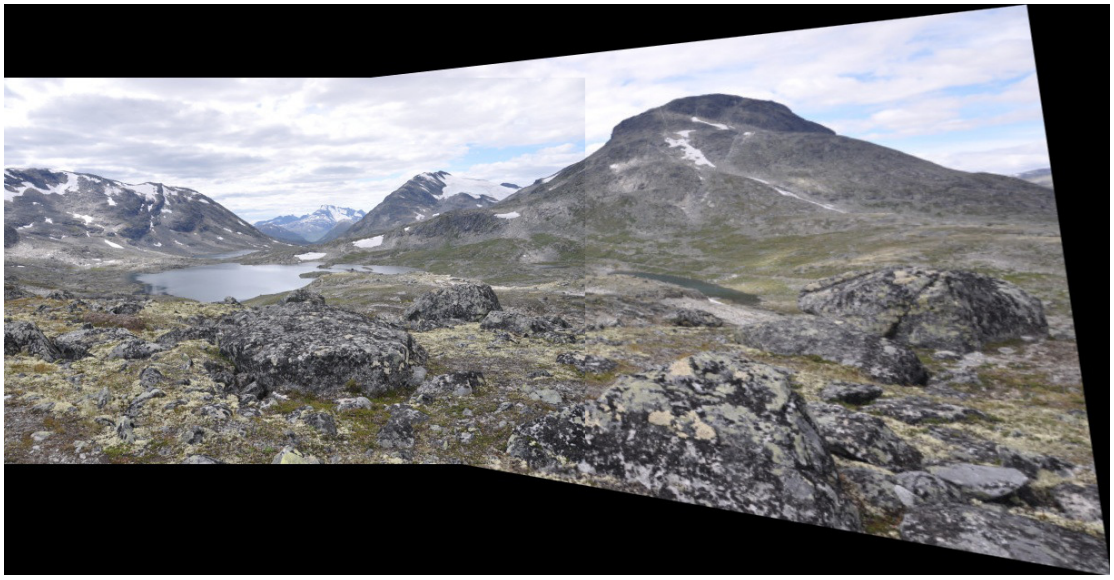
Ukončující kritérium algoritmu RANSAC (počet iterací) je definováno rovnicí 2.6. Parametr p je jediný, který můžeme ovlivnit. Rozhodl jsem se tedy provést několik testů a zjistit, jaký má tento parametr vliv na výsledné panorama.

Při testovaní jsem sledoval počet nalezených inliers a napojení fotografií na sebe (kvalitu výsledných homografií). Hodnocení napojení fotografií bylo určováno zkoumáním panorama, kde jsem si určil hraniční napojení fotografií (drobné nedostatky v napojení fotografií jsou přehlíženy). Příklad hraničního panorama, které je prohlášeno za dobré, naleznete na obrázku 4.4. Hodnoty v tabulce 4.2 ve sloupci inliers jsou průměrná čísla z padesáti testů.

p	**inliers**	**dobrých panoramat**
0,95	71,74	45
0,75	69,24	39
0,55	67,24	38
0,35	62,70	32
0,15	60,52	29

Tabulka 4.2: Výsledky experimentů. Hodnoty ve sloupci dobrých panoramat udávají počet, kolik panoramat z padesáti bylo prohlášeno za dobré.

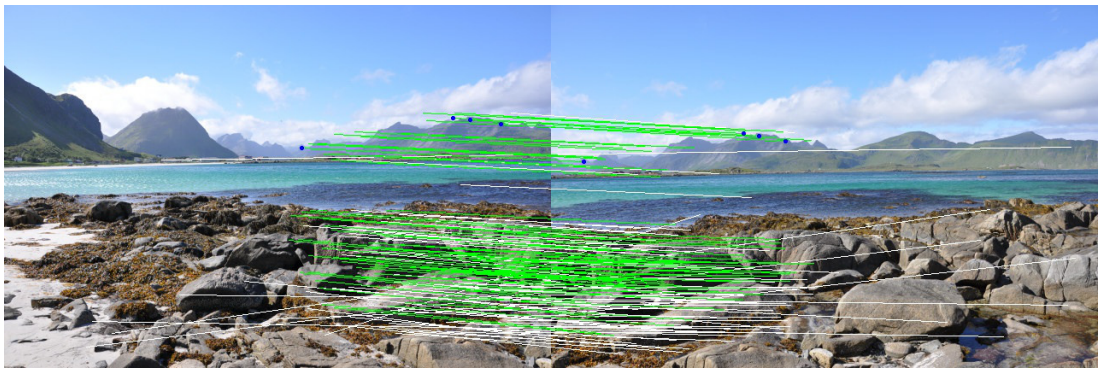
Na základě hodnot v tabulce můžeme provést následující závěry:
Ve více jak polovině případů je správná homografie vypočítána během prvních několika iterací (v mém případě se jednalo o deset a méně iterací). Velmi zajímavé je, že po snížení parametru p na hodnotu 0,75 dojde k nejrazantnějšímu poklesu dobrých panoramat. Je to způsobeno poklesem provedených iterací náhodných výběrů skoro na polovinu. Ve sloupci inliers jsem očekával větší rozdíly hodnot. Tyto hodnoty potvrzují, jak je důležité rozložení inliers korespondencí po co největší oblasti překrývajících se částí fotografií.



Obrázek 4.4: Hraniční panorama prohlášené za dobré

4.3 Nedostatky programu

Při testování jsem se také setkal se situací, kde panorama s méně korespondencemi prohlášenými za inliers dosáhla lepšího výsledného napojení než panorama s více korespondencemi prohlášenými za inliers. Na obrázcích 4.5 a 4.6 můžeme vidět důvod, proč tomu tak je. Homografii odpovídá více inliers, ale všechny jsou nadetekovány pouze v dolní části fotografií (obrázek 4.6). Pokud je homografie ohodnocena více inliers, neznamená to ještě, že výsledná homografie provede lepší spojení fotografií. Tento problém by šlo vyřešit vybíráním zdrojových bodů pro výpočet homografie z různých oblastí společné části fotografií. V programu jsem ale tuto problematiku neřešil.



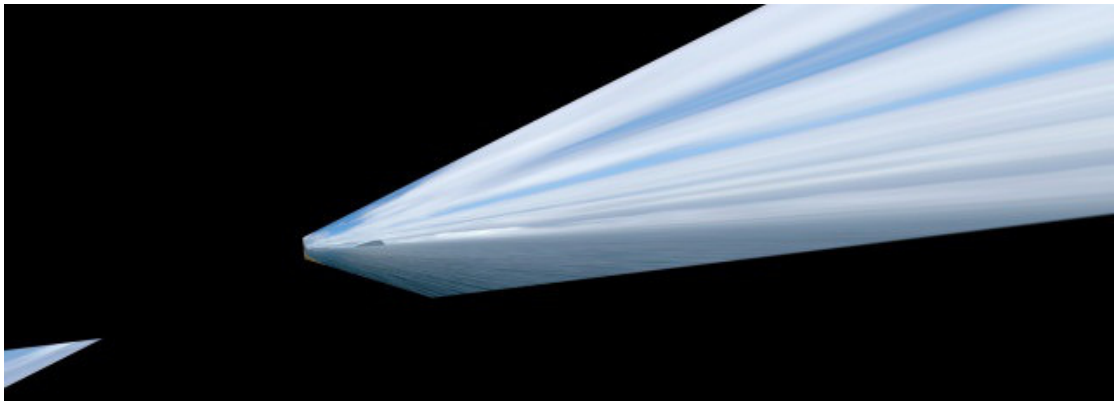
Obrázek 4.5: Předběžné korespondence, 80 inliers (zvýrazněno zeleně)

Při testovaní s různými zdrojovými fotografiemi jsem narazil na problém u jedné dvojice. S těmito fotografiemi program vykazuje velmi špatné výsledky. Dochází k častému výpočtu nevhodných homografií, což má za následek nevhodné napojení fotografií. Ukázku takovéhoto napojení nalezneme na obrázku 4.7. V některých případech dochází k výpočtu tak špatné homografie, že od určité pozice se pravý horní roh začne mapovat úplně mimo panorama (dojde k roztržení zdrojové fotografie, obrázek 4.8). Tyto chyby budou pravděpodobně způsobeny nevhodnými zdro-

Obrázek 4.6: Předběžné korespondence, 90 inliers (zvýrazněno zeleně)

jovými fotografiemi. Fotografie byly pořízeny širokoúhlým objektivem z velmi blízké vzdálenosti. V popředí došlo k velkému perspektivnímu zkreslení útesu, zatímco v pozadí skoro k žádnému. U obou zdrojových fotografií není nalezeno mnoho význačných bodů ve skále v pozadí. V části, kde se fotografie překrývají je tedy pouze pár korespondencí. Podobná situace nastává v překrývající se části v popředí (malé množství korespondencí v této oblasti). Největší část korespondencí je v malé oblasti (okolí „drápu“ skály). Když sečteme tyto nedostatky, dostaneme pravděpodobně příčinu, proč u těchto fotografií dochází k výpočtu chybných homografií.

Obrázek 4.7: Chybně vytvořené panorama



Obrázek 4.8: Chybná transformace pravé fotografie, pravý horní roh je odtržen od panorama. Tento snímek není výstupem programu, ale byl vytvořen speciálně ve fázi hledání příčiny chyby.

4.4 Shrnutí algoritmů

RANSAC je velmi starý algoritmus (poprvé představen roku 1981 [2]). Z hlediska porovnání se zbylými dvěmi algoritmy nepřináší žádné výhody, ale oba algoritmy z něho vychází. Proto se jedná o velmi důležitý algoritmus.

LO-RANSAC (s jednoduchou optimalizací) umožňuje zpřesnění výsledné homografie a to má za následek lepší napojení fotografií na sebe. Toto zpřesnění má za následek delší výpočetní čas. V mých testech se konkrétně jednalo skoro o dvojnásobný čas než jaký potřeboval algoritmus RANSAC. Je třeba dodat, že i přes toto časové navýšení se výpočet pohybuje v desítkách ms.

PROSAC je výpočetně nejméně náročný algoritmus a to i díky nejmenšímu počtu iterací náhodného výběru. Algoritmus je stabilnější než RANSAC (nachází lepší homografie a nachází je častěji).

Kapitola 5

Závěr

S ohledem na stále zvětšující se popularitu fotografovaní na celém světě, tedy i odvětví tvorby panoramatických fotografií, považuji téma této práce za velice zajímavé. A jelikož se tímto oborem fotografie sám zabývám, práce pro mě byla velmi přínosná.

Hlavní náplní této práce nebylo skládání panoramat jako celek, ale pouze jedna část tohoto postupu (výpočet homografie). V této problematice jsem si zvolil tři algoritmy, které se zde využívají a provedl jsem jejich porovnaní z různých hledisek. Z hlediska problematiky skládání panoramat je program značně limitován, ale pro porovnání zvolených algoritmů plně postačuje.

Pro detekci bodů jsem využil algoritmus SURF, který kromě významných bodů také poskytuje popisy těchto bodů (deskriptory). Díky tomuto faktu se zjednodušuje i tvorba vstupních dat pro algoritmy na výpočet homografie, kde využívám porovnaní deskriptorů pro určení korespondencí. Implementace tohoto detektoru je součástí knihovny OpenCV.

U výpočtu homografie mě z porovnávaných algoritmů nejvíce zaujal algoritmus PROSAC, který je velmi nenáročný na výpočetní čas a podává kvalitní výsledky. Ještě zajímavějších výsledků by tento algoritmus mohl podávat ve spojení s jednoduchou lokální optimalizací.

V rámci pokračování práce v budoucnosti se nabízejí dva směry:

1. Rozšířit práci o další algoritmy pro výpočet homografie, kde existuje mnoho dalších zajímavých algoritmů.

2. Provést zdokonalení ve směru skládání panoramat. Znamenalo by to hlavně tvoření panoramat z více než dvou fotografií a zrušení aktuálního předpokladu zadávání zdrojových fotografií v pořadí jak na sebe navazují. Dále také využití speciálního algoritmu pro zobrazení překrývajících se části fotografií.

Výhodou rozšíření práce ve směru dokonalejšího generátoru panoramat, je řešení nových problematik. Proto shledávám toto rozšíření za zajímavější a přínosnější.

Literatura

[1] *OpenCV Wiki* [online]. Poslední modifikace: 1. dubna 2011. [cit. 30. dubna 2011]. Dostupné na: <http://opencv.willowgarage.com/wiki/>.

[2] *RANSAC* [online]. Poslední modifikace: 17. dubna 2011. [cit. 20. dubna 2011]. Dostupné na: <http://en.wikipedia.org/wiki/RANSAC>.

[3] BARKER, K. *Singular Value Decomposition Turtorial*. [b.m.]: University of the Witwatersrand, 2005. 24 s. Výzkumná zpráva. Dostupné na: <http://www.cs.wits.ac.za/ michael/SVDTut.pdf>.

[4] BAY, H., ANDREAS, E., TUYTELAARS, T. et al. *Speeded-Up Robust Features (SURF)*. [b.m.]: ETH Zurih and K.U. Lueven, 2008. 14 s. Výzkumná zpráva. Dostupné na: <ftp://ftp.vision.ee.ethz.ch/publications/articles/eth_biwi_00517.pdf>.

[5] CHUM, O. a MATAS, J. *Locally Optimized RANSAC*. [b.m.]: Czech Technical University of Prague, 2005. 7 s. Výzkumná zpráva. Dostupné na: <http://cmp.felk.cvut.cz/ matas/papers/chum-prosac-cvpr05.pdf>.

[6] CHUM, O., MATAS, J. a KITTLER, J. *Locally Optimized RANSAC*. [b.m.]: Czech Technical University of Prague and University of Surrey, 2008. 8 s. Výzkumná zpráva. Dostupné na: <http://cmp.felk.cvut.cz/ matas/papers/chum-dagm03.pdf>.

[7] G. LOWE, D. *Distinctive Image Features from Scale-Invariant Keypoints*. [b.m.]: University of Brirish Columbia, 2004. 28 s. Výzkumná zpráva.

[8] GEORGIA COLLEGE OF TECH COMPUTING. *RANSAC for fitting translations and homographies* [online]. říjen 2004. Dostupné na: <http://www.cc.gatech.edu/classes/AY2005/cs4495_fall/assignment4.pdf>.

[9] KRŠEK, P. *Základy počítačové grafiky: Studijní opora předmětu IZG* [online]. [cit. 20. dubna 2011]. Dostupné na privátních stránkách predmětu IZG.

[10] NAVRÁTIL, J. *Transformace a RANSAC: Prezentace z přednášky předmětu POV* [online]. [cit. 20. dubna 2011]. Dostupné na veřejných stránkách predmětu POV.

[11] WINDISCH, G. *Scale Invariant Feature Transform*. [b.m.]: University Koblenz-Landau, 2006. 9 s. Výzkumná zpráva. Dostupné na: <http://www.uni-koblenz.de/~gwindisch>.

Příloha A

Uživatelský manuál

Pro správný chod programu je potřeba knihovna OpenCV 2.1. Návod na instalaci této knihovny nalezneme například v literatuře [1]. Navržený program je konzolová aplikace. Program se spouští s následujícími parametry:

bp -RANSAC/-LORANSAC/-PROSAC <left_image> <right_image> [<panorama_name>]

kde:

-RANSAC/-LORANSAC/-PROSAC určuje jaký algoritmus chceme použít pro výpočet homografie

<left_image> je název levé zdrojové fotografie včetně přípony (případně cesta)

<right_image> je název pravé zdrojové fotografie včetně přípony (případně cesta)

[<panorama_name>] je nepovinný parametr s názvem výsledného panorama bez přípony, pokud tento parametr není zadán, panorama se uloží s názvem panorama

Program podporuje skládání panoramat pouze ze dvou zdrojových fotografií, které je potřeba zadat v pořadí jak na sebe navazují.

V případě, že nejsou zadány správné zdrojové fotografie (neexistující, nenavazující fotografie, případně jsou zadáný ve špatném pořadí), program vypíše nápovědu pro správné spuštění a ukončí se. Výstupem programu je počet nalezených inliers, počet iterací algoritmu pro výpočet homografie, čas vypočtu algoritmu a v novém okně se zobrazí výsledné panorama, které se také uloží ve stejném adresáři, kde byl program spuštěn, s příponou jpg. Program se ukončuje stiskem libovolné klávesy.

Příloha B

CD

Struktura CD je následující:

/documentation	programová dokumentace
/latex	zdrojové soubory pro LaTeX
/openCV	instalační soubory knihovny openCV
/program	spustitelný program a testovací obrázky
/source	zdrojové soubory programu
/text	práce ve formátu pdf